**AFRL-IF-RS-TM-2007-5**
**Final Technical Memorandum**
**February 2007**


# HIGH ORDER NON-STATIONARY MARKOV MODELS AND ANOMALY PROPAGATION ANALYSIS IN INTRUSION DETECTION SYSTEM (IDS)

**Advanced Technical Concepts**

*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.*

**STINFO COPY**

**AIR FORCE RESEARCH LABORATORY**
**INFORMATION DIRECTORATE**
**ROME RESEARCH SITE**
**ROME, NEW YORK**

# NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the Air Force Research Laboratory Rome Research Site Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-IF-RS-TM-2007-5 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

| | |
|---|---|
| /s/ | /s/ |
| ANNA L. LEMAIRE<br>Work Unit Manager | IGOR G. PLONISCH, Chief<br>Strategic Planning & Business Operations Division<br>Information Directorate |

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

# REPORT DOCUMENTATION PAGE

*Form Approved*
**OMB No. 0704-0188**



| 1. REPORT DATE *(DD-MM-YYYY)* | 2. REPORT TYPE | 3. DATES COVERED *(From - To)* |
|---|---|---|
| FEB 2007 | Final | May 06 – Sep 06 |

**4. TITLE AND SUBTITLE**

HIGH ORDER NON-STATIONARY MARKOV MODELS AND ANOMALY PROPAGATION ANALYSIS IN INTRUSION DETECTION SYSTEM (IDS)

**5a. CONTRACT NUMBER**

**5b. GRANT NUMBER**
FA8750-06-1-0176

**5c. PROGRAM ELEMENT NUMBER**
62702F

**6. AUTHOR(S)**

Victor A. Skormin

**5d. PROJECT NUMBER**
558B

**5e. TASK NUMBER**
II

**5f. WORK UNIT NUMBER**
RS

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Advanced Technical Concepts, Inc.
352 Ford Hill Road
Berkshire NY 13736-2135

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

AFRL/IFB
26 Electronic Parkway
Rome NY 13441-4514

**10. SPONSOR/MONITOR'S ACRONYM(S)**

**11. SPONSORING/MONITORING AGENCY REPORT NUMBER**
AFRL-IF-RS-TM-2007-5

**12. DISTRIBUTION AVAILABILITY STATEMENT**
*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 07-074*

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**

A new concept targeted to decrease false positive rates of anomaly based intrusion detection operating in the system call domain is proposed. To mitigate false positives, network based correlation of collected anomalies from different hosts is suggested, as well as a new means of host-based anomaly detection. The concept of anomaly propagation is based on the premise that false alarms do not propagate within the network. Unless anomaly propagation is observed, alarms are to be treated as false positives. The rationale behind the concept lies in the fact that the most common feature of worms and viruses is self-replication. As replication takes place, a malicious code propagating through the network would carry out the same activity resulting in almost identical system call sequences and triggering the same alarm at different hosts. The alarm propagation effect can be used to distinguish "true alarms" from "false positives". At the host-level, a new anomaly detection mechanism operating that employs non-stationary Markov models is proposed.

**15. SUBJECT TERMS**
False positive rates, intrusion detection, anomaly detection, worms, viruses, self-replication, malicious code, non-stationary Markov models

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | Anna L. Lemaire |
| U | U | U | UL | 13 | 19b. TELEPHONE NUMBER (*Include area code*) |

**Standard Form 298 (Rev. 8-98)**
**Prescribed by ANSI Std. Z39.18**

# HIGH ORDER NON-STATIONARY MARKOV MODELS AND ANOMALY PROPAGATION ANALYSIS IN IDS

Victor A. Skormin vskormin@binghamton.edu

## Final Report on Grant FA8750-06-1-0176


### ABSTRACT

A new concept targeted to decrease false positive rates of anomaly based intrusion detection operating in the system call domain is proposed. To mitigate false positives, network based correlation of collected anomalies from different hosts is suggested, as well as a new means of host-based anomaly detection.

The concept of anomaly propagation is based on the premise that false alarms do not propagate within the network. Unless anomaly propagation is observed alarms are to be treated as false positives. The rationale behind the concept lies in the fact that the most common feature of worms and viruses is self-replication. As replication takes place, a malicious code propagating through the network would carry out the same activity resulting in almost identical system call sequences and triggering the same alarm at different hosts. The alarm propagation effect can be used to distinguish "true alarms" from "false positives".

At the host-level, a new anomaly detection mechanism operating that employs non-stationary Markov models is proposed. Many applications or services have different operating modes, which have different dynamics with respect to system call issuance. Therefore, an application or service can be treated as a non-stationary stochastic process and be modeled with a non-stationary Markov chain, which significantly improves model consistency compared to stationary Markov chain.


### INTRODUCTION

The first Intrusion Detection System (IDS) utilizing system calls was proposed in [1]. Today, these systems utilize two main approaches, misuse detection and anomaly detection. Misuse or signature-based detection systems utilize descriptions of known attack expressed in terms of system calls. Although signature-based systems can provide high level of accuracy, they fail to detect previously unknown attacks. Anomaly detection systems utilize models of normal behavior of legitimate processes, especially privileged ones. These systems check the consistency between the invoked system calls and the profile of normality for a given process and have the potential to detect unknown attacks, though they frequently suffer from a high rate of false positives.

This research targets anomaly-based IDSs that in spite of their advantages are impractical due to high rate of false positives. The limited success of known research aimed at the alleviation of this problem [2, 3, 6, 8] is due to it being primarily aimed at the improving the accuracy of the normality models (profiles) rather than achieving high confidence in classifying the detected anomaly.

Two major contributions of this paper are as follows. First, a novel host-level anomaly detection mechanism is proposed. Second, having efficient host-level anomaly detection, the unique but rather simple principle, *false positives do not propagate*, is suggested as the basis for establishing, with high degree of confidence, whether detected anomaly is a false positive or a true positive.

The anomaly detection mechanism utilizes non-stationary Markov models. While many shell codes and exploits (in buffer overflow attack) may use only 20-30 system calls, which would certainly be concealed in a histogram, Markov models are clearly preferable to other order insensitive techniques (such us frequency histograms) used to model normality profiles [1, 3]. However, the common assumption that the source (application or service) is a stationary stochastic process generally may not by true. Any application or services utilize high level functions intended to solve different tasks. When an application realizes several related tasks or group of tasks which

condition each other, it is supposed to operate in one of its distinct phases (modes). For example, our preliminary experiments have identified the following major operation modes of Internet Explorer: application loading, browsing (loading pages from the Internet) and downloading (retrieving large files).

These operation phases are distinguished by functionality and achieve different goals. Since different operation modes would have their own realization with respect to system calls, it can be assumed that operation phases have different unconditional as well as conditional distribution of system calls. Hence, the system calls profile of an application or service should be modeled as a non-stationary stochastic process.

The so-called “moving omnibus” method is used to distinguish bifurcation points (points of sudden change in dynamics) in observed data, which would confine its stationary phases, then use obtained phases to train Markov models. As a result each process would have set of Markov models corresponding to each operation phase, which would certainly increase model consistency.

It is expected to dramatically decrease false positives by correlation of anomaly reports from different hosts in the network. The main distinguishable feature of viral software is self-replication. It is differently implemented in viruses and worms and can be revealed by the detection of specific (abnormal) sequences of system calls [4]. As the self-replication continues, the propagation of the same abnormal activity pattern could be observed within the network. We call it the anomaly propagation.

The utilization of system call attributes provides unambiguous representation of the connectivity between various computers and processes within the network. Then, if the anomaly propagation pattern is consistent with the process connectivity pattern, it could be declared with a high degree of certainty that the detected anomaly is *true positive*, otherwise it is *false positive*.

The proposed IDS approach has two levels of implementation, the host-level anomaly detection, and the network-level attack detection.

## RELATED WORK

Signature-based IDS utilizing system call data are known in literature. The feasibility of anomaly detection using system calls is shown in [1], [2], [3]. The efficiency of this approach can be enhanced further by the analysis of system call attributes as shown in [5], [6], [7], [8] and [9]. Additional improvement of this approach was demonstrated in [4]. The misuse detection-based IDS approach could be best exemplified by [10], [11].

## PROPOSED ANOMALY DETECTION

Traditionally, anomaly detection consists of recognizing process behavior deviation from a profile of normalcy. In the system calls domain an IDS compares sequences of system calls against a model and consider abnormal traces as anomalous. The efficiency of the anomaly detection depends on model accuracy. The best way to model an application in the system calls domain is to derive a system call execution graph, which would explicitly reflect all possible braches in algorithms. However, it is impossible to process all possible branches of initial algorithms from the binary code due to the implicit logic transitions (jumps). Thus one can only derive only approximate model, which adds some degree of uncertainty in the application description. To reflect such uncertainty, it can be assumed that system calls are emitted by a stochastic source (an application) with categorical state space represented by system calls.

Any application or services utilize high level functions these functions are intended to solve different tasks. When an application realizes several correlated tasks or group of tasks that condition each other, it is supposed to operate in one of its distinct phases (modes). These operation phases are distinguished by functionality and achieve different goals. Since different operation modes would have their own realization with respect to system calls, it can be assumed that operation phases would have different unconditional as well as conditional distribution of system calls. Hence, system calls profile of an application or service can not be modeled as stationary stochastic process, but as non-stationary stochastic process.

Operation phases consist of many system calls and implement some strictly prescribed high-level tasks. This consideration assumes the source to be stationary over each operation phase. Since dynamics of the

stochastic process are appeared to be invariant over an operation phase, we can model operation phases by Markov chains. Therefore, the source would be modeled by set of Markov models corresponding to each operation phase.

In this context, the trace of system calls is considered to be anomalous if it is not likely to happen according to current Markov model (corresponding to current operation phase). The sequence is not expected to happen if it was not predicted by Markov model. Therefore, the anomaly score can be chosen as prediction performance of the Markov model over the observed sequence. Prediction performance can be represented by chi-square likelihood ratio of the observed sequence of certain window. If chi-square likelihood ratio exceeds specified threshold we declare observed sequence as anomalous.

## OPERATION PHASE DETECTION

Before deriving Markov models, operation phases must be distinguished automatically in unsupervised fashion. We have to apply a method which for the given sequence of observations (system calls) to determine bifurcation points (moments of dramatic change in process dynamics). These bifurcation points would certainly correspond to moments of operation phase switching.

One of the most efficient techniques for detecting bifurcation points is moving "omnibus" method [15]. The method is simple extension of rather classical "omnibus" method. The latter one uses Pearson's $\chi^2$ hypothesis test. The observed sequence of states (system calls) ($S = \{s_1,...,s_n\}$) is partitioned by two contiguous subsequences ($S_1 = \{s_1,...,s_k\}$ *and* $S_2 = \{s_{k+1},...,s_n\}$) for some dividing point $k$. Then, every subsequence ($S_1, S_2$) is used as training set to compute corresponding transition matrix ($T_1, T_2$). We can state the following test to check if *k* is bifurcation point:

$$H_0 : T_{1,2}(i,j) = T(i,j), \quad versus \quad H_1 : T_{1,2}(i,j) \neq T(i,j) \qquad (1)$$

where, *T* is global transition matrix computed over the entire sequence *S*

If the null hypothesis is rejected then transitional probabilities are time variant due to dynamic change in point *k*. Therefore, rejection of $H_0$ points out that *k* is bifurcation point with some degree of confidence. To implement the test (1) we can compute test statistic in the following way:

$$W = \sum_{k=1}^{2} \sum_{i,j=1}^{m} \left( N_1(i) \frac{(T_1(i,j) - T(i,j))^2}{T(i,j)} + N_2(i) \frac{(T_2(i,j) - T(i,j))^2}{T(i,j)} \right)$$

where $N_1(i), N_2(i)$ - marginal observed frequency of i-th state derived form first and second subsequences respectively.

By central limit theorem, statistic $W$ is asymptotically distributed as $\chi^2$ with $2(m-1)^2$ degree of freedom under null hypothesis. Hence, we can compute p-value of the test in the following way: $p_{value} = 1 - F_\chi(W)$ , where $F_\chi - \chi^2\ cdf$ . If p-value lower than chosen test size $\alpha$, we reject $H_0$ and claim non-stationarity with $|\alpha - p_{value}|$ significance.

Moving "omnibus" method consists of deriving p-value for test (1) with respect to center point of the window sliding over the observed sequence (trace). The sliding window must be long enough to derive local Markov models from the left and right halves of the window. The points rejecting null hypothesis would be declared as bifurcation points. Having set of bifurcation points we can chose the most appropriate ones according to constraints for phase minimum length and number of phases. The algorithm for selecting such points is proposed below.

Input constraints:

$n$ –number of locally stationary phases

$l_{\min}$ – minimum length of a phase

Algorithm:

1. Form list $L$ of sorted bifurcation points in decreasing order with respect to significance.
2. Take first $n$ points from list $L$ and compute length of phases enclosed by these points
3. For every phase, which length is less than $l_{\min}$ delete from the list $L$ less significant boundary point.
4. Continue step 2 until all constrained are met or there is no enough points left in the list.

If the process has enough locally stationary periods, the algorithm will determine bifurcation points enclosing these periods in the observed sequence.

To demonstrate the efficiency of the algorithm on a real process, the sequence of system calls was

observed during two operation phases of the Internet Explorer. The first phase consists of browsing different sites without downloading large files and the second phase was downloading large files. Figure 1 depicts results of the algorithm. The plot shows p-value for every separated point moved from 5000 instance to the end of the sequence (40000). The size of hypothesis testing was 5 percent. It could be seen that the test was rejected only once in the separating point 15000 what shows high accuracy of bifurcation point detection. P-value changed dramatically in earlier points (13000-14000), but never reached significance level. The results of the experiment show the high efficiently of the moving "omnibus" method.

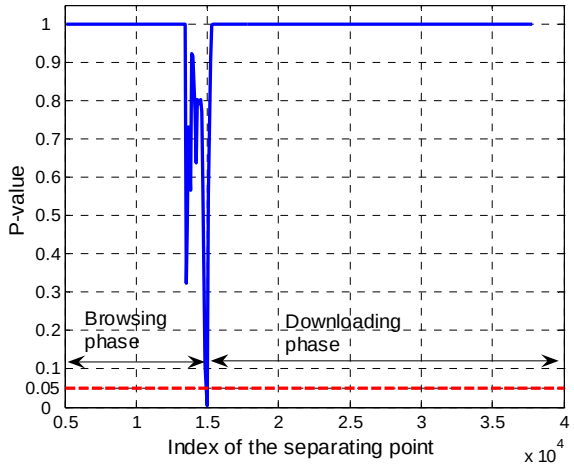


**Figure 1 Bifurcation point detection**

Having facility to recognize stationary operation phases in observed sequence of system calls, these segments (subsequences) of states (system calls) can be used to derive Markov models in offline. Nevertheless, in testing regime, it is necessary to recognize (in real time) which phase the monitored application operates in to apply corresponding Markov model. The problem of matching current outcomes to set of models was addressed in several publications (eg Stolfo [3]). Many authors use simple approach consisting in trying different models form the set and choosing the one which best fits according to some distance metric. We used the similar method, if current Markov model is not consistent according to predicting performance metric (likelihood ratio), the system searches for the model having the greatest performance. To avoid undesired frequent model change we introduced a constraint on minimum number of system calls before model switching is allowed.

## MARKOV MODELS ORDER DETECTION

System calls are invoked according some algorithms which have logical structure and solves its own tasks which constitute operation phases. Since system calls are issued in consecutive logic order, the probability of occurrence of system calls depends on not only one previous system call, but several preceding system calls (prefix of system calls). These considerations lead to necessity of using high order Markov models.

The order of the Markov chain corresponding to an operation phase is not known, hence has to be determined. There are several approaches to determine order of Markov model consistent with observed sequence. These methods include: transition correlation based method, chi-square statistics of the transition frequency, index of transition complexity and information theoretic approach.

We used information theoretic approach to determine proper order of Markov model. The rationale beside the approach is the fact that if probability of occurrence of a state highly depends on $n$ previous states then mutual information of the current state and n preceding states must be greater than information based on less than $n$ past states. The criteria for defining best order of the Markov model can be formally presented in the following way:

1. Starting from $n=1$
2. Compute n-gram transition probabilities: $p(i \mid \underline{i_1,...,i_n})$

   where, $p(i,\underline{i_1,...,i_n})$ probability of occurring state $i$ given past prefix of states $i_1,...,i_n$ (order preserving prefix)
3. Compute $n+1$ and $n$ order mutual information:

   $$I(X_{n+1};X_1,...,X_n)=\sum_{i_1,...,i_{n+1}\in 1..N} p(i_{n+1},..,i_1)\log\left(\frac{p(i_{n+1},..,i_1)}{p(i_{n+1})p(i_n,...,i_1)}\right)$$

   $I(X_n;X_1,...,X_{n-1})$ - to be computed analogically
4. D= $I(X_{n+1};X_1,...,X_n)$ - $I(X_n;X_1,...,X_{n-1})$
5. If $D<\tau$ then stop, otherwise $n=n+1$ and go to step 2

This algorithm will derive the maximum order of the model which still provides specified mutual information increase. In reality mutual information rate is decreasing function of the model order for given observation sequence. Our preliminary

experiments showed that phases have at most third order models.

## ANOMALY REPLICATION DETECTION

The main distinguishable feature of activity perpetrated by such malicious software as worms or viruses is self-replication. Viruses have to create file containing its copy or attach itself to some victim file. In contrast, worms may not even create any files, residing in memory space of the victim process and performing some activity on behalf of victim, legitimate process. Thus, worms may not leave any file "traces", which can be used to detect the fact of self replication. Nevertheless, to perform some activity, injected malicious code must utilize system resources trough system calls what will certainly reflect in process behavior and detected as anomaly. Replication assumes that worm will copy itself to memory space of the remote process. Thus, every copy of the worm would carry out the same activity what would result in the same system calls trace and almost the same detected anomaly in the victim process. Therefore, self-replication can be revealed by propagation of system calls traces (detected anomaly sequences of system calls) from process to process residing in different hosts. We call this approach **anomaly propagation**.

The detected abnormal subsequence issued by the same worm may contain also legitimate system calls, which can be explained by inertia of the anomaly detector. *Thus abnormal sequences may loose some (not many) "alien" system calls in the prefix and have short suffix of legitimate system calls invoked by victim process after getting the execution control back.*

To distinguish real propagation from set of coincidences (meaningless occurrence of the same anomaly in different nodes of the network) connectivity pattern of the nodes (processes) is analyzed. If propagation pattern is consistent with connectivity pattern between processes, anomaly replication takes place, which is a result of malicious activity. If anomaly propagation is not consistent with connectivity pattern, a false positive is declared.

Anomaly propagation analysis requires distinguishing similar abnormalities to consider them as single anomaly. Anomalous sequences of system calls detected on the hosts are reported to the server provided with time stamp and source process ID. In the server the anomalies are stored and grouped. Anomaly sequences in one group represent the same anomaly. When server receives the abnormal sequence it searches for the closest group. The distance between two anomalies is reciprocal of similarity measure which is represented by length of the longest common factor. We can formalize the distance between a sequence ($S$) and a group of sequences ($G$) in the following way:

$$d(S,G) = \min_{\tilde{S} \in G} \left( \frac{|S|}{|c(S,\tilde{S})|} \right) \qquad (2)$$

where, $c(S,\tilde{S})$ - the longest common factor (substring) of $S$ and $\tilde{S}$, $|S|$ - length of the sequence $S$. If the distance to the closest group is less than specified threshold, we consider the new anomaly the same as anomalies from the group and add it to the group. If the distance is exceeds the threshold, we treat new anomaly as previously unseen and contribute it to a new group as the first member.

The rationale behind using the proposed distance lies in the fact that processes may be attacked and subverted at any time. Thus we can expect that anomaly sequences would have different short suffixes (segment of legitimate system calls). Subsequence from the beginning to the legitimate suffix would be large segment forming the pure sequence of system calls actually invoked by worm's payload. Hence, anomalous sequences cased by the same worm will have large common factor which would constitute some segment of "malicious" system calls.

Longest common factor between two strings (with length n, m) can be found through dynamic programming with $O(n \cdot m)$ computational complexity. Hence, determining the closest group by one to one search would exhibit $O(N \cdot m)$ complexity, where $N$ – sum of the length of strings in the cumulated anomaly dictionary. However, we do not need to know the longest common factor itself to compute the distance $d(S,G)$, but only the length of the factor. The length of the $c(S,\tilde{S})$ can be approximated by length of longest common subsequence which can be found trough weighted Levenshtein (edit) distance for the case when $c(S,\tilde{S})$

constitutes significant part in both sequences. The longest common subsequence is not necessary contiguous, but due to high performance of the proposed anomaly detector, we can expect that "alien" part in anomaly substring (common factor) would be much longer than the rest (legitimate) what justifies approximation. The edit distance is weighted so that, deletion operations would not constitute any penalty score. Formally, the length of the longest common factor of sequences $S$ and $\tilde{S}$ can be represented in the following way:

$$\begin{cases} \left|c(S,\tilde{S})\right| = |S| - L_W(S,\tilde{S}) \\ L_W(S,\tilde{S}) = L(S,\tilde{S}) - L_{delete} \\ L(S,\tilde{S}) = L_{sub} + L_{insert} + L_{delete} \end{cases} \quad (3)$$

where, $L(S,\tilde{S})$ - Levenshtein (edit) distance between $S$ and $\tilde{S}$, $L_{sub}, L_{insert}, L_{delete}$ - number of substitution, insertion and deletion operations respectively. Here $L_W(S,\tilde{S}) = L_{sub} + L_{insert}$ represent weighted edit distance with zero penalty of deletion operation.

Expressions 2 and 3 shows that minimizing $L_W(S,\tilde{S})$ we will minimize the distance $d(S,G)$. The problem of finding the closest anomaly group can be reformulated in the following way: given the dictionary of strings (sequences) $D$ and a pattern string $S$ find the string $\hat{S}$ from dictionary which is the closest to $S$ with respect to weighted edit distance $L_W(S,\tilde{S})$. Having the closest string $\hat{S}$, we consider the distance (2) as distance to the group the string $\hat{S}$ belongs to.

Such problem is called approximate dictionary querying and is addressed in several papers [12, 13, 14]. Yates and Navarro [12] use metric property of edit distance (triangular inequality) to neatly organize vocabulary as a metric space. Such data structure reduces dictionary query complexity to $O(N \log N)$. Brodal and Gasieniec [13] utilized cell-probe model to achieve very low complexity. Nevertheless the method handles only one mismatch queries ($L_W(S,\tilde{S}) \le 1$) what is not applicable to our problem (length of the suffix could be more than 1). We suggest using method presented by Cole and Lewenstein [14] which uses so called longest common prefix data structure to organize vocabulary. The method can handle cases with edit distance more than one and has query complexity $O\left(m + \frac{(c \log d)^k}{k!} \log\log N\right)$ which is less than $O(N \log N)$ and $O(N \cdot m)$ for $k, m << n$, where $d$ – number of strings in the dictionary, $k$ – specified maximum (preferred) distance in the query, c – some constant. Since, legitimate prefix is expected to be small in anomalous sequence, preferred edit distance $k$ would be also small. The only modification in the algorithm [14] concerns objective function, which must be changed to weighted edit distance depicted in expression (3).

After defining the closest group of anomalies, the system must analyze the pattern of propagation of anomalies in the group to reveal replication feature. Replication property is determined through processes connectivity pattern. Connectivity pattern (or connectivity graph) is represented by weighted directed graph $C(V,E)$ with nodes being processes in different hosts and edges presenting last interaction. If one process sends some data to the port another process listens to, we assume that sender process interacted to the second one. Thus, if one process 1 interacted to the process 2, the system adds edge (or upgrades if there is already one) with weight equal to relative interaction time.

Anomaly propagation is considered to have replication pattern if it is consistent with connectivity graph in both topological and time sense. In other words, if anomaly is replicating, it must propagate according to simple rules:

- Each new instance of anomaly (except the first one) must occur in the process which has recently been interacted by another suspicious process (which already issued anomaly)
- The time elapsed from last interaction and anomaly occurrence must not be longer than prescribed threshold (active window)

For the multipartite attack (coordinated multi source malicious activity) the first rule must tolerate several sources. Iterative algorithm verifying if anomalies propagation in the group has replication pattern would be straightforward:

**Input**

**Ne**w anomaly $V_A$ being added to the group $G = \{V_1,..,V_n\}$

Weighted adjacency matrix $T$ of the connectivity graph $C$ $T(i, j) = weight(E(V_i, V_j))$

- Get the subset of group members connected to anomalous process:
  $S = \{V : (V \in G) \& (T(V, V_A) > 0)\}$
- Check if the last connection time is less than threshold: $\min_{V \in S}(T(V, V_A) < t_{\max}$
- Increase counter of the group members participating in the replication $k = k + 1$
- If the normalized counter is more than threshold ($\frac{k}{|G|} > \tau$), arise attack alarm
- If group size is more than prescribed value and normalized counter is less than tolerance value $\left(\frac{k}{|G|} < tol\right) \& \left(|G| > s_{\max}\right)$, declare false positive.

This algorithm provides score which is compared to threshold to decide if anomaly propagation is indeed replication. The score takes into account the relative number of instances matched to replication pattern and shows how much the propagation similar to replication pattern. If the size of the group exceeded some size threshold and score is still pretty low system will declare false positive. Since only false positive may have many instances (unknown operation phase massively turned in many hosts) and not have propagation pattern.

## EXPERIMENTAL RESULTS

We performed trace based simulation using recorded system calls were from legitimate processes as well as malicious software. As malicious agent, we choose forth generation of Sasser worm – Sasser.D worm. Simulated legitimate processes include Microsoft Internet Explorer and CCAPP. We do not claim comprehensive monitoring in these preliminary experiments, however size of records constituted tens of thousands. For instance, we used 50000 system calls issued by Internet Explorer for crating Markov models. We also recorded 24 contiguous system calls invoked by victim process executing Sasser worm payload.

Using the system call records, we obtained non-stationary models for three chosen processes. Figure 2 depicts call prediction performance for CCAPP process based on stationary Markov model versus non-stationary model. Non-stationary model contains three dynamic invariant chains. The trends present chi square likelihood ratio statistic which formally reflects prediction performance, the lower statistic the better prediction. Examining the curves we can see that non-stationary Markov model (solid line) totally outperforms stationary model (dashed line).

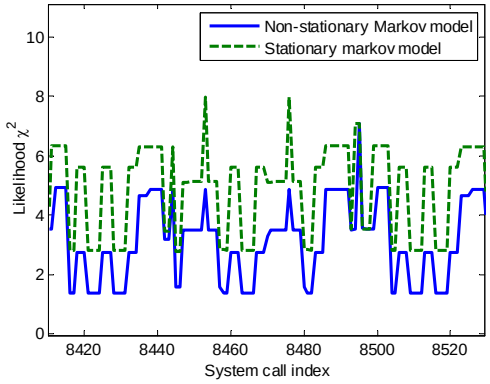


**Figure 2 Predicting performance for CCAPP**

After obtaining Markov models, we performed discrete time simulation in MATLAB. In the beginning of the simulation, 100 fictitious homogenous processes, running on 100 imaginary hosts, are assigned random starting index of system call in the trace (pool) of the recorded calls from some fixed legitimate process. Now we assume that processes start execution form that assigned position. For sufficiently large trace, it is reasonable to assume that any starting point distribution may happen in real life. In other words, if the trace of a process is really long, then this process in any host will eventually pass through some segment in the trace during corresponding operation phase in real life.

Simulated processes running in 100 assumed hosts virtually invoke system calls from the general trace one after one starting from assigned position. Delay period between two subsequent invoked calls is randomly generated after every invoking. This helps to reflect different delays of system calls execution due to operation system overload, what frequently happens in real life especially during massive viral attack. Moreover, to simulate connectivity between processes every process establishes connection to randomly chosen another process with some stochastically changed periodicity.

Attack is simulated through inserting Sasser's segment in front of the current trace position of target

process. The target process virtually invokes the worm trace and continues invoking system calls from the legitimate trace, what reflects normal execution return in real life. Attack pattern is specified prior and is performed with corresponding virtual inter-process connections.

Sasser.D worm is indented to attack LSASS process, but today the vulnerability is well known and any antivirus software can detect it. Since system calls invoked in the penetrated process does not reflect exploit itself, but payload, we assumed that the same payload may be used in other attacks for different process. Thus, we decided simulate attack to Internet Explorer. On the other hand, we choose Internet Explorer for simulation because it the most complicated process with respect to modeling it in system call domain. It uses many system calls and have several operation phases, moreover it is multithread application what presumably decrease model accuracy making it the most challenging for attack detection.

We simulated slow worm attack, which is difficult to detect. Attack pattern had tree-like shape. 30 different attacks were performed. All of them were detected in early stages and no false positive (falsely declared attack) observed.

We present results of detection of one of the attacks. Propagation score of the attack is depicted in the figure 3. Figure shows 12 groups arranged in score descending order. First group contain indexes of attacked hosts. One can see that other “normal” groups have score five times lower than the score of the first group. The attack was detected on fourth instance of process subverting, what shows agility of the detection scheme in spate of the lots of noise

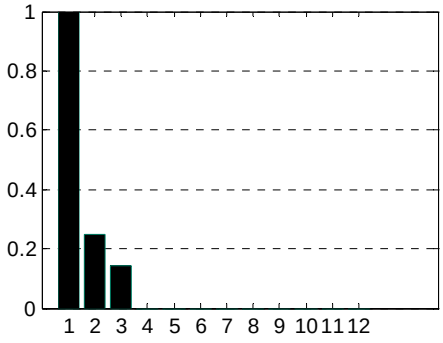


**Figure 3 Propagation score**

caused by random connectivity simulation and legitimate anomalies.

Figure 4 shows local false positive rate of those four processes which participate in attack pattern. This false positives shows host based detection without propagation analysis, decision made on hosts before sending anomaly to server. One can see that for threshold less then 20 all of the hosts have high local false positive. We repeated in offline the same attack but with local thresholds ranging from 10 to 20. In all cases, attack was successfully detected and no global false positive observed, what shows robustness of the anomaly propagation detection.

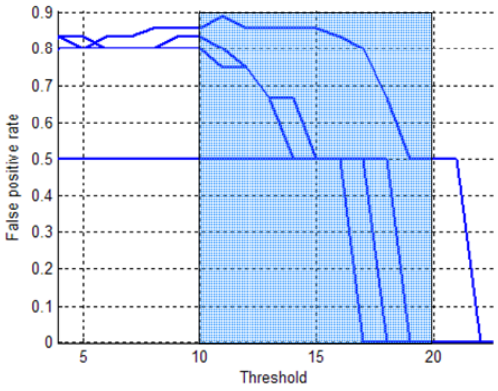


**Figure 4 False positive rate for different thresholds**

Figure 5 shows predicting performance of one of the attacked hosts. The big wide spike is due to Sasser trace. In this case of course worm anomaly has high score, but likelihood ratio depends on model quality what may end up in good prediction for worm trace (in case of bad model) and as a result in false negative of host based detection. One can see that for threshold 15 stride dashed line, even if the statistic would be truncated down to shaded region (small dashed) what is three times less, the worm trace would anyway be detected locally and sent to the server. And as it was mentioned above attack for local threshold from 10 to 20 (including 15) was successfully detected. This result shows that even for bad model the network level detection can still reveal the attack

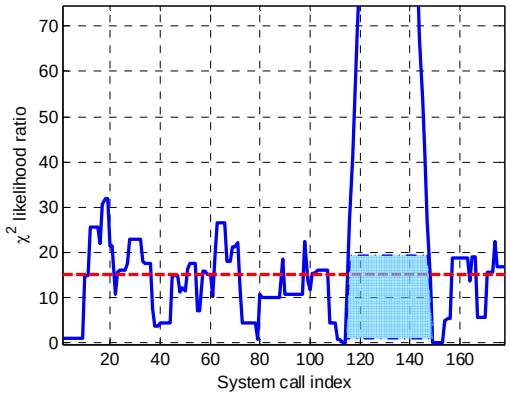


**Figure 5 Anomaly score (predicting performance)**

## CONCLUSION

We have demonstrate that anomaly propagation concept along with non-stationary Markov models can provide a high level of confidence in detection; no misdetections were observed in preliminary experiments. Non-stationary Markov models uniformly outperform stationary Markov models. Anomaly replication is detected for wide range of local threshold. Low threshold for local anomaly score (chi square likelihood ratio) leads to large number of detected local anomalies what constitutes impending noise for propagation detection. Nevertheless, even for low thresholds, anomaly propagation was reliably detected in all experiments, what points out high robustness and dependability of the proposed concept.

Future work will be focused on problems white list application and extensions for multipartite (many sources) attack detection. White list means that declared false positive anomalies will be transmitted to the hosts to make filters of false positive. The main question is how big the white list would be until it saturates. Which distance threshold we should choose to declare if a new anomaly is indeed contained in the list.

Multipartite attack must be treated in the little different way and may not have tree-like propagation pattern. Propagation concept will be generalized to handle multi source attack patterns.

## REFERENCES

[1] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of Computer Security*, vol. 6, no. 3, pp. 151–180, 1998.

[2] A Durante, R Di Pietro, LV Mancini. "Formal Specification for Fast Automatic IDS Training". *Lecture Notes in Computer Science,* 2629**:**191-204, 2003

[3] SJ Stolfo, W Lee, E Eskin. "Modeling system calls for ID with Dynamic Window Sizes", In *Proceedings of the DISCEX II*. June 2001.

[4] V. Skormin, A. Volynkin, D. Summerville, J. Moronski, **"**Run-Time Detection of Malicious Self-Replication in Binary Executables" *Journal of Computer Security*, 2006 in appearing

[5] Alexander Liu, Cheryl Martin. "A Comparison of System Call Feature Representations for Insider Threat Detection", In *Proceedings of the 6th IEEE Information Assurance Workshop,* 2005

[6] Gaurav Tandon, Philip K. Chan. "Learning Useful System Call Attributes for Anomaly Detection". In *Proceedings of the FLAIRS Conference*, 2005

[7] Bowen, T., Segal, M., and Sekar, R. "On preventing intrusions by process behavior monitoring". In *Proceedings of the Workshop on Intrusion Detection and Network Monitoring*, 1999

[8] Xu M, Chen C, Ying J. "Anomaly detection based on system call classification". *Journal of Software*, 15(3): 391~403, 2004

[9] Christopher Kruegel, Darren Mutz, Fredrik Valeur and Giovanni Vigna. "On the Detection of Anomalous System Call Arguments". *8th European Symposium on Research in Computer Security (ESORICS), Lecture Notes in Computer Science, Springer Verlag Norway* October 2003.

[10] Bernaschi, M., Grabrielli, E., Mancini, L. "Operating System Enhancements to Prevent the Misuse of System Calls", In *Proceedings of the ACM Conference on Computer and Communications Security*, 2000

[11] Kang, D.-K., Fuller, D., and Honavar, V. "Learning classifiers for misuse and anomaly detection using a bag of system calls representation". In *Proceedings of 6th IEEE Systems Man and Cybernetics Information Assurance Workshop (IAW).* 2005

[12] R. Baeza-Yates, G. Navarro. "Fast approximate string matching in a dictionary", In *Proceedings of the SPIRE'1998 IEEE Computer Press*, 1998

[13] G. S. Brodal and L. Gasieniec. “Approximate dictionary queries”. In *Proceedings of the Symposium on Combinatorial Pattern Matching*, 65-74, 1996.
[14] R. Cole, Lee-Ad Gottlieb, M. Lewenstein. “Dictionary Matching and Indexing with Errors and Don't Cares”, *Annual ACM Symposium on Theory of Computing*, 2004
[15] J. M. Gottman, Roy A Kumar. “Sequential analysis. A guide for behavioral researchers”, *Cambridge: Cambridge University Press*, 1990
[16] V. Skormin, A. Volynkin, D. Summerville, J. Moronski, “Prevention of Information Attacks by Run-Time Detection of Self-Replication in Computer Codes,” Acepted for publication in *Computer Security Journal*
[17] A. Volynkin, V. Skormin, D. Summerville, J. Moronski, "Evaluation of Run-Time Detection of Self-Replication in Binary Executable Malware," *7th Annual IEEE Information Assurance Workshop*, West Point, NY, June 6, 2006